UML Representation of Extended Role-Based Access Control Model with the Use of Usage Control Concept

Aneta Poniszewska-Maranda

UML representation of extended role-based access control model with the use of usage control concept

Aneta Poniszewska-Maranda

Institute of Information Technology, Technical University of Lodz, Poland
anetap@ics.p.lodz.pl

Abstract. This paper presents an extension of role-based access control model with the use of usage control concept together with its representation using the Unified Modeling Language (UML). The presented model is developed for role engineering in the security of information system. The presented implementation of URBAC (Usage Role-Based Access Control) model consists in creation of security profiles for the users of information system.

1 Introduction

Recently, rapid development in different technologies of information systems have caused the computerizing of many applications in various business areas. Data has become very important and critical resource in many organizations and therefore efficient access to data, sharing the data, extracting information from the data and making use of the information has become an important and urgent necessity.

Access control is a significant part of each information system. It is concerned with determining the allowed activities of system users and mediating every attempt by a user to access a resource in the system. The evident objective of access control is to protect the system resources from any undesired user accesses. Nowadays, the access control is connected with the great development of information technologies and methodologies that allow to create more complex, dynamic information systems.

Data protection against improper disclosure or modification in the information system is the important issue of each security policy realized in the institution. Access control policies, strategies or models should also be changed to manage the security requirements of modern information systems (e.g. to manage the dynamic aspects of access control), such as dispersion of data and resources, dynamic changes both in data proprieties and users' proprieties, responsibilities and abilities from the point of view of access control rules, dispersion of users, complex system organization with different users, rules and security principles that are sometimes mutually exclusive. On the other hand, distributed information systems or federation of information systems provide the access of many different users to huge amount of data, sometimes stored in different locations

and secured by different strategies, security policies and models or inner enterprise rules. These users have different rights to the data according to their business or security profiles that depend on their organization positions, actual locations and many other conditions. The system data is transferred between the particular nodes of distributed system.

It is also important to protect the information against non-controlled utilization and control the usage and diffusion of the information. It gives the possibility to specify how it can be used and specify the utilization constraints. It seems, the new mechanisms of access control security should be defined to apply for distributed information systems.

Compared to the traditional models, URBAC approach assures the usage control in data accessing that is very important, especially in distributed information systems, and the organization of the access control strategies well-described in RBAC (Role-Based Access Control) model or its extensions. In the paper we propose the new access control approach for dynamic, complex information systems that additionally provides the common coherence of information system components on the global level of access control strategy.

This paper presents an extension of the role-based access control model with the use of usage control concept together with its representation using the Unified Modeling Language (UML). The presented model is developed for role engineering in the security of information system. The presented implementation of RBAC model consists in creation of security profiles for the users of information system. The entire procedure is performed in two stages: defining the permissions assigned to a function and providing the definitions of functions assigned to a particular role.

The paper is structured as follows: section 2 presents the related works on access control concepts, section 3 deals with security requirements of dynamic information systems and describes the extension of access control model for dynamic information systems. Section 4 shows the representation of URBAC approach using the UML concepts while section 5 describes the rules for production of roles based on URBAC approach.

2 Role concept and usage concept in access control

The development of access control policies and models has a long history. It is difficult to mention all the access control models that were specified in literature and this is not the objective of this paper. It is possibly to distinguish two main approaches. The first one represents the group of traditional access control models. Discretionary Access Control (DAC) model [15, 17], the first model in these group, manages the users access to information based on user identification and on the rules defined for each user (i.e. subject) and each object in the system using the access control matrix. However, the DAC model has the inherit weakness that information can be copied from one object to another and it is difficult for DAC to enforce the safety policy and protect the data against some security attacks.

In order to prevent the shortcomings of DAC model, the Mandatory Access Control (MAC) model was created to enforce the lattice-based policies [15]. In this model each subject has their own authorization level that permits them the access to objects starting from the classification level, which has lower or equal range. MAC does not consider the covert channels but they are expensive to eliminate. Next, Sandhu et al. proposed the Role-Based Access Control (RBAC) model [1, 2] that has been considered as an alternative to DAC and MAC models. This model requires the identification of roles in a system. The RBAC model was a progress in access control but it is still centered around the access control matrix and has static character, particularly from the point of view of distributed information systems [1].

The second approach of access control models corresponds to the temporal models that introduce the temporal features into traditional access control. The temporal authorization model was proposed by Bertino and al. in [18] that is based on the temporal intervals of validity for authorization and temporal dependencies among authorizations. Next, the Temporal-RBAC (TRBAC) model was proposed in [19]. This model introduces the temporal dependencies among roles. Other model - Temporal Data Authorization model (TDAM) was presented in [20] and extends the basic authorization model by temporal attributes associated to the data such as transition time or valid time. Recently, the TRBAC model was extended to Generalized Temporal RBAC (GTRBAC) model in [21] to express the wider range of temporal constraints.

Currently, traditional access control models are not sufficient and adequate in many cases for information systems, especially modern, dynamic, distributed information systems, which connect different environments by the network. Some disadvantages of these models in security domain were found [12]:

- traditional access control models provide only the mechanisms for definition of authorizations but do not give the possibility to define the obligations or conditions in the access control,
- access rights can be only pre-defined by the developers or security administrators and granted to the subjects,
- decision about the access can be only made before the required access, not during the access,
- it is not possibly to define the mutable attributes for subjects and for objects.

These disadvantages and the needs of present information systems caused the creation of unified model that can encompass the use of traditional access control models and allow to define the rules for dynamic access control. Two access control concepts are chosen in our studies to develop the new approach for dynamic information systems: role concept and usage concept. The first one allows to represent the whole system organization in complete, precise way while the second one allows to describe the usage control with authorizations, obligations, conditions, continuity (ongoing control) and mutability attributes.

Role-Based Access Control (RBAC) [1, 2] requires the identification of roles in a system. The role is properly viewed as a semantic structure around which

the access control policy is formulated. In *extended RBAC (eRBAC)* model [10, 11] each role realizes a specific task in the enterprise process and it contains many functions that the user can take. For each role it is possible to choose the necessary system functions. Thus, a role can be presented as a set of functions that this role can take and realize.

The *Usage Control (UCON)* [7–9] is based on three sets of decision factors: authorizations, obligations and conditions that have to be evaluated for the usage decision. The obligations and conditions are added in the approach to resolve certain shortcomings characteristic for the traditional access control strategies.

3 Access control approach for dynamic information systems

Actual information systems can contain many different components, applications, located in different places in a city, in a country or on the globe. Each of such components can store the information, can make this information available to other components or to different users. The authorized users accessing the information can change this information, its status, role or other attributes at any time. These changes can cause the necessity of modifications in security properties of accessed data on access control level. Such modifications are dynamic and often should be realized ad hoc because other users from other locations can request the access to the information almost at the same time. Many different users from different locations can access information system. Sometimes, they need to have the direct and rapid access to actual information. However, the conditions of such access are very often dynamic, they can change for example because of actions of other users. It is necessary to ensure the ad hoc checking of current rights of an user basing on their current conditions that can change dynamically.

An example can be a health-care system that contains the sensitive information and should provide secure access to highly sensitive patient information over Internet. Patient records can be modified by a primary physician or by a specialist and almost at the same time can be used by the receptionist to schedule the consultation or special treatment. Some patient records, new or modified, can be sensitive and closed to other users. Decision to allow the access to these records or not should be taken ad hoc basing on new attributes or modified attributes related with information that can change dynamically depending on the accesses of authorized users and its operations and transaction of these data.

Therefore, there is the need to have the access control approach that will describe the organization that should be secured, their structure in proper and complete way, like RBAC model does, and on the other hand it will be appropriate and sufficient for dynamic information system, like usage control concept.

Usage Role-based Access Control approach The proposed access control approach is based on two access control concepts: role concepts and usage concepts. It was named Usage Role-based Access Control (URBAC).

The core part of URBAC approach essentially represents the extended RBAC model (Fig. 1). We distinguished two types of users in URBAC: single user (*User*) and group of users (*Group*). These two elements are represented by the element **Subject** that is the superclass of User and Group. *Subjects* can be regarded as individual human beings. They hold and execute indirectly certain rights on the objects [22].

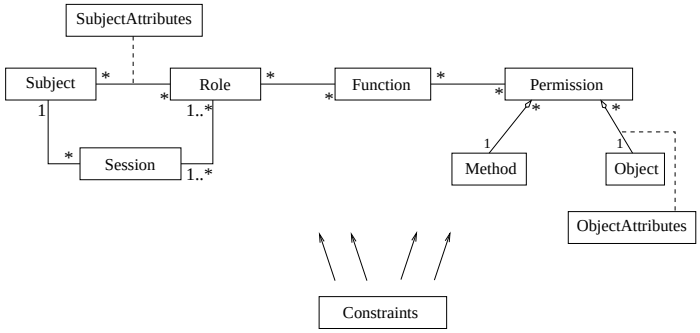

Fig. 1. Elements of Usage Role-based Access Control

Subject permits to formalize the assignment of users and groups to the roles. Subject can be viewed as the base type of all users and groups in a system. It can be presented as an abstract type, so it can not have direct instances - each subject is either a user or a group. A **User** is a human being, a person or a process in a system, so it represents the system entity, that can obtain some access rights in a system. **Group** represents a group of users that have the same rights. Subjects can be assigned to the groups by the aggregation relation *SubjectGroup* that represents an ordering relation in the set of all system subjects. Groups of users are often created in enterprise management information systems as PLM systems or ERP systems to provide the possibility to assemble a set of people in a group with the same obligations, responsibilities or privileges (e.g. persons that realize the same business project in an enterprise).

The **Session** element represents the period of time during which the user is logged in a system and can execute its access rights. In our model the *Session* is assigned directly to the *Subject*, i.e. an individual user or a user from a group is login into information system during a single session. On the other hand a session is connected with the *roles* and this association represents the roles that can be activated during one session.

A **Role** is a job function or a job title within the organization with some associated semantics regarding the authority and responsibility conferred on a member of the role. The role can represent a competency to do a specific task, and it can embody the authority and responsibility. The roles are created for various job functions in an organization.The users are assigned to the roles, based on their responsibilities and qualifications. The direct relation is established be-

tween roles and subjects that represent the users or groups of users. The user can take different roles on different occasions and also several users (*Group* element) can play the same role. It is also possible to define the hierarchy of roles, represented by aggregation relation *RoleHierarchy*, which presents the inheritance relations between the roles. Hierarchy of roles represents also the inheritance relations between the roles. The role of the part end of the relation inherits all privileges of parent role. The association relation between the roles and subjects is described by the association class **SubjectAttributes** that represents the additional subject attributes (i.e. subject properties) as in usage control. *Subject attributes* provide additional properties, describing the subjects, that can be used for the usage decision process, for example an identity, enterprise role, credit, membership, security level.

Each role allows the realization of a specific task associated with an enterprise process. A role can contain many functions **Function** that a user can apply. Consequently, a role can be viewed as a set of functions that this role can take to realize a specific job. It is also possible to define the hierarchy of functions, represented by relation named *FunctionHierarchy*, which provides the hierarchical order of system functions. Hierarchy of functions, just like hierarchy of roles, represents also the inheritance relations between the functions. The function of the part end of the relation inherits all privileges of the parent function.

Each function can perform one or more operations, a function needs to be associated with a set of related permissions **Permission**. A function can be defined as a set or a sequence (depending on particular situation) of permissions. To perform an operation one has the access to required object, so necessary permissions should be assigned to corresponding function. Therefore, all the tasks and required permissions are identified and they can be assigned to the users to give them the possibility to perform the responsibilities involved when they play a particular role. Due to the cardinality constraints, each permission must be assigned to at least one function to ensure the coherence of the whole access control schema [8].

The permission determines the execution right for a particular method on the particular object. In order to access the data, stored in an object, a message has to be sent to this object. This message causes an execution of particular method **Method** on this object **Object**. Very often the constraints have to be defined in assignment process of permissions to the objects. Such constraints are represented by the authorizations and also by the obligations and/or conditions.

Authorization (A) is a logical predicate attached to a permission that determines the permission validity depending on the access rules, object attributes and subject attributes. **Obligation (B)** is a functional predicate that verifies the mandatory requirements, i.e. a function that a user has to perform before or during an access. They are defined for the permissions but concerning also the subjects - *Subject* can be associated with the obligations which represent different access control predicates that describe the mandatory requirements performed by a subject before (*pre*) or during (*ongoing*) an access. **Conditions (C)** evaluate the current environmental or system status for the usage decision

concerning the permission constraint. They are defined also for the permissions but they concern the session - *Session* can be connected with the set of conditions that represent the features of a system or application.

A constraint determines that some permission is valid only for a part of the object instances. Therefore, the *permission* can be presented as a function *p(o, m, Cst)* where*o* is an object, *m* is a method which can be executed on this object and *Cst* is a set of constraints which determine this permission. Taking into consideration a concept of authorization, obligation and condition, the set of constraints can take the following form $Cst = \{A, B, C\}$ and the permission can be presented as a function $p(o, m, \{A, B, C\})$. According to this, the permission is given to all instances of the object class except the contrary specification.

The **objects** are the entities that can be accessed or used by the users. The objects can be either privacy sensitive or privacy non-sensitive. The relation between objects and their permissions are additionally described by association class **ObjectAttributes** that represents the additional object attributes (i.e. object properties) that can not be specified in the object's class and they can be used for usage decision process. The examples of object attributes are security labels, ownerships or security classes. They can be also mutable or immutable as subject attributes do.

The **constraints** can be defined for each main element of the model presented above (i.e user, group, subject, session, role, function, permission, object and method), and also for the relationships among the elements. The concept of constraints was described widely in the literature [5, 6, 13, 14, 16]. It is possible to distinguish different types of constraints, static and dynamic, that can be attached to different model elements.

The URBAC distinguishes the following general types of constraints:

- *Authorizations* - constraints defined for the permissions, basing on access rules defined by enterprise security policy but also basing on objects' attributes and subjects' attributes. The authorizations evaluate the subject attributes, object attributes and the requested permissions together with a set of authorization rules for the usage decision. Authorizations can be either *pre-authorizations* (decision is made before the access) or *ongoing-authorizations* (decision is made during the access).
- *Obligations* - constraints defined for the permissions but concerning also the subjects - *Subject* can be associated with the obligations which represent different access control predicates that describe the mandatory requirements performed by a subject before (*pre*) or during (*ongoing*) an access. These requirements should be verified before or during an access realized by a user. They can represent the security constraints that are defined on the subjects (i.e. users or groups) and they can be static or dynamic. The obligations can be *pre-obligations* (utilize a history function to check if the certain activities have been fulfilled or not) or *ongoing-obligations* (have to be satisfied continuously or periodically while the allowed rights are during a usage).
- *Conditions* - constraints defined also for the permissions but they concern the session - *Session* can be connected with the set of conditions that repre-

sent the features of a system or application. They can describe the current environmental or system status and states during the user session that are used for the usage decision. There are two types of *conditions*: *pre-condition* (checked before a usage and *on-condition* (has to be satisfied while a usage). The example of conditions can include current time for accessible time period (e.g. business hours), current location of a user for accessible location data (e.g. enterprise area) or system security status (e.g. normal, alert, attack). The conditions mainly focus on evaluations of environmental, system-related restrictions that have no direct relationships with subjects, users and accessible data for the usage decision. The subject or object attributes can be used to select which condition requirements have to be used for a request. The evaluation of conditions cannot update any subject or object attributes.

- Constraints on roles and on functions. The most popular type in this group of constraints are *Separation of Duty (SoD)* constraints [5, 11].
- Constraints on relationships between the model elements [5, 11, 12].

The detailed view of usage role-based access control approach with the set of all elements and relationships presented above is given on figure 2.

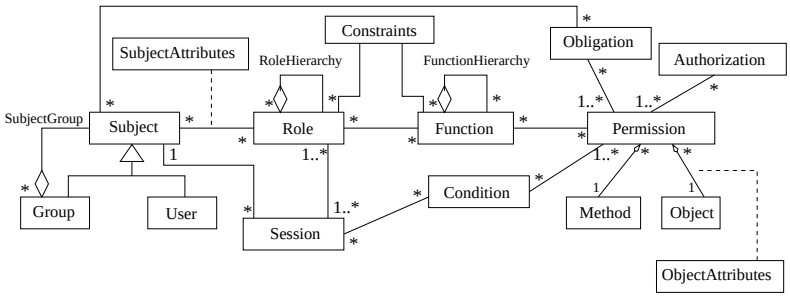

Fig. 2. Meta-model of URBAC model

4 Representation of URBAC with the use of UML concepts

Currently, Unified Modeling Language (UML) is a standard language for analysis and design of information systems [3, 4]. It is widely known and used in software engineering field to support the object oriented approach. UML gives the possibility to present the system using different models or points of view. It has a set of diagrams to represent the elements of whole information systems or one of its components. Some chosen features of UML can be used to implement URBAC approach, especially during the design of information system and its associated security schema based on URBAC.

Therefore, UML can be used in role engineering process to implement and realize URBAC approach. To accomplish this, the concepts of UML and URBAC should firstly be joined. Two types of UML diagrams have been chosen to provide the URBAC: use case diagram and interaction diagram. The use case diagram presents the system's functions from the user point of view. It define the system's behavior without functioning details. According to UML meta-model, each use case from use case diagram should be described by a scenario and in consequence by at least one interaction diagram (i.e. sequence diagram or communication diagram). The interaction diagram describes the behavior of one use case [3, 4]. It represents the objects and messages exchanged during the use case processing.

The relationships between UML concepts and concepts of usage role-based access control are presented below (Fig. 3).

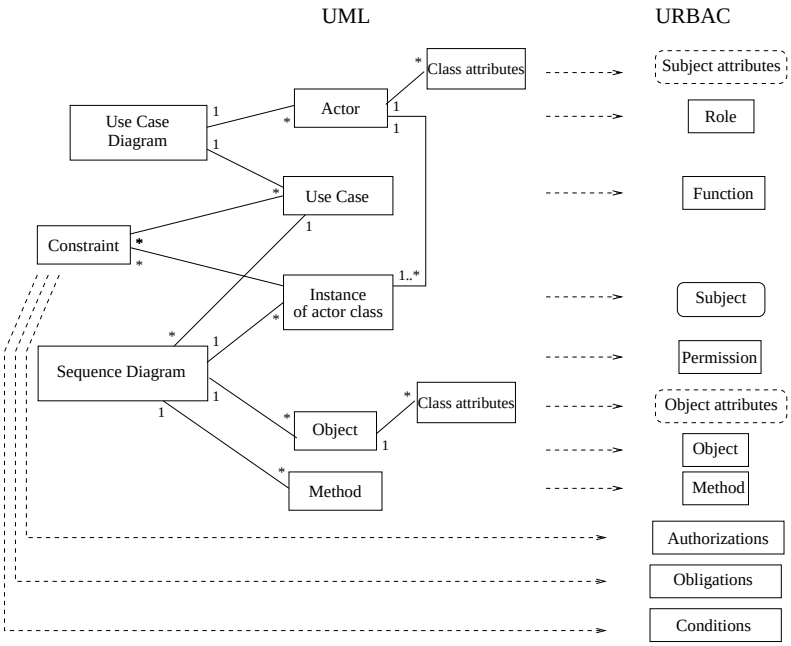

Fig. 3. Relationships between concepts of URBAC and UML concepts

4.1 Elements of URBAC and corresponding UML concepts

UML describes the processes that occur in a system or application and such processes are use case oriented. Use case is the main concept of UML used in analysis and design of software. It is used for specifying required usages of a system, to capture the requirements of a system, describes what a system is supposed to do.

This approach is realized with the use of use case diagram that is one of two the most important diagrams in the design process. Use case diagram contains the concept of an actor that is very close to the role concept of RBAC model. UML defines an actor as a coherent set of roles playing by the users of an use case during the interaction with this use case or in general with a system. Therefore, a role from access control can be presented as an UML actor.

Each actor cooperates with one or more use cases representing the system's functions. Each use case specifies some behavior, possibly including variants, that the system can perform in collaboration with one or more actors. These behaviors, involving interactions between the actors and the system, may result in changes to the state of the system and communications with its environment. Thus, a function from URBAC can be represented by an UML use case. As it was presented in the previous section, a set of functions can be determined for each role. Similar situation exist in UML design - each actor can be in interaction with a set of use cases and these relations specify the relations of R-F type (between roles and functions).

In UML approach each use case from use case diagram should be described by scenario and in consequence by at least one interaction diagram (sequence diagram or communication diagram). Although, both the interaction diagrams return the similar results, the sequence diagrams gives more possibilities from version 2.0 of UML. The concept of sequence diagram has been chosen for this reason (it allows to represent the remaining elements of URBAC).

Sequence diagram represents an interaction of objects by means of a sequence of messages sent between these objects. In each sequence diagram there is one or more objects representing the actors who can in this way participate directly in the process of sending the messages between the objects. It is possible to distinguish different types of messages sent to objects in sequence diagram. One of them represents a method call. The methods executed in sequence diagram and also in other UML diagrams can represent the methods of URBAC. Similarly, the objects that occur in UML diagrams, e.g. sequence diagram, collaboration diagram, can be attached to the object concept of access control model.

Access to the object, that is the possibility of execution of method on an object, is controlled with respect to the right of the method execution possessed by a message sender (i.e. subject) over an object. The access model allows for each subject the specification of a list of methods that the subject can execute. Therefore, for each use case it is necessary to specify the permissions for method's execution and as it is shown above these permissions can be found examining the sequence diagram describing the particular use case (that represents the URBAC function). These permissions are assigned to the functions by the function-permission (F-P) relation.

The relations between the elements of URBAC can be found also by the examination of use case diagram and interaction diagram (in particular sequence diagram). A use case diagram offers four types of relations between its elements:

- communication relation between an actor and a use case that represents the relation between a role and a function, i.e. R-F relation,

– generalization relation between actors, representing the inheritance relation between roles (R-R relation),
– two types of relations between use cases represent the inheritance relations between functions of URBAC, i.e. F-F relations: including relation (with stereotype *include*) corresponds to standard inheritance relation and extending relation (with stereotype *extends*) corresponds to inheritance relations with additional constraints that specify the conditions of these extensions.

The second important diagram of UML, besides the use case diagram, is class diagram. This type of UML diagrams is used to describe the types of objects in a system and their relationships. Set of attributes and set of operations can be defined for each class in class diagram.

Each actor specified in the use case diagram can be also defined in details in form of UML class on the class diagram. The system user (i.e. person) is one of actor's types, thus a user is an instance of actor class. The set of class attributes forms the attributes characteristic for this actor. Therefore, the subject attributes (e.g. user attributes) from URBAC can be represented by the set of attributes defined for an instance of actor class of UML.

According to UML meta-model, each system's object has to be of particular type, thus new objects in a systems can be created only for classes defined in class diagram. Each class has among other the set of attributes. Therefore, the concept of object attributes from URBAC can be attached to set of attributes defined for this object in its class specification.

4.2 Constraints of URBAC and corresponding UML concepts

The concept of constraints in URBAC approach corresponds directly to the constraint concept existing in UML. The security constraints of URBAC can be defined for different elements and for relations between these elements. These constraints can be presented on UML diagrams corresponding to types and locations of elements for which these constraints will be defined with the use of OCL (object Constraint Language) that is the part of UML approach.

Five types of security constraints were identified for URBAC approach (Section 3). The authorization is a constraint attached to a permission that determines the permission validity depending on defined access rules. It can be represented by the UML constraint defined for the method's execution in sequence diagram.

The obligation is a constraint defined on a permission but it concerns also the subject (e.g. a user) - subject should fulfill the obligation executing a function before or during an access. This type of constraints can be presented as UML constraint in sequence diagram (as pre-condition or an invariant), especially from version 2.0 of UML that provide the combinated fragments in sequence diagrams (e.g. "alt" or "opt") allowing the definition of constraint conditions.

The UML constraints representing the authorizations or obligations can be also supported by relations between the elements concerning these constraints.

The condition is a constraint also defined on a permission but it concerns the session element. It defines current environmental or system status and states during the user session that can be used for the usage decision. The conditions can be also represented by UML constraints defined in sequence diagrams (mainly as an invariants).

Remaining types of constraints represent the constraints defined for the roles, functions and their relations. Such constraints are represented by the UML constraints defined for actors, use cases and their relations on use case diagrams or sometimes on sequence diagrams.

5 Production of roles based on URBAC approach

Security policies of a system generally express the fundamental choices taken by an organization to protect the data. They define the principles on which access is granted or denied. Role-based access control models are highly effective models, especially in large companies that have high turnover rates due to the fact of allowance for the administrator to simply place the new employees into the roles instead of creating the new permissions for each new person who joins the company and needs to be included in the system.

However, on the other hand, access control mechanisms still demand clear definition of set of activities and operations for each system's user that he will be allowed to execute. Consequently, a set of permissions should be defined for the user. As it was shown above, a set of permissions composes indirectly the system's roles. Therefore, the production of roles using the URBAC approach consists in determination of permissions for the functions and next functions for the application/system role with the consideration of defined security constraints (e.g. authorizations, obligations, etc.).

The process of role production can be automatic or partially automatic and is based on the connections between UML and URBAC, described in section 4. This process is realized with the use of use case diagrams, where system roles and functions are defined and with the use of sequence diagrams, where permissions are assigned to the rights of execution of methods realized in each use case. These two types of diagrams should be examined to identify the roles of URBAC, the functions that are used by these roles to interact with the information system and the permissions needed to realize these functions. First of all the rules for creation of set of roles were defined.

5.1 Rules for creation of roles

Each subject (i.e. user or group of users) in an information system is assigned to a security profile (i.e. user profile) which is defined by the set of roles that can be played by him. A security profile is defined by a pair *(s, listRoles(s))*: *s* is a subject, *listRoles(s)* is a set of roles assigned to this subject. Taking into consideration the concept of user, such profile can be defined as follows: *(u, listRoles(u))*, where *u* is a user, *listRoles(u)*.

It is possibly to give the rules of role creation and the assignments of these roles to users or groups of users:

1. A access control profile should be defined for each subject (i.e. user) who interact with the system

$$s_i \vdash securityProfile_{s_i}$$

or in particular for an user:

$$u_i \vdash securityProfile_{u_i}$$

2. This profile is defined by a set of roles which can be assigned to the subject (i.e. user) with the respect of subject attributes defined mainly on level of security administrator

$$securityProfile_{s_i} \vdash setRoles_{s_i}, subjectAtt_{s_i}$$

in particular for an user:

$$securityProfile_{u_i} \vdash setRoles_{u_i}, subjectAtt_{u_i}$$

To receive a significant profile, each subject (i.e. user) should be assigned at least to one role.

3. A role is defined by a set of functions assigned to this role with respect of potential security constraints defined for them

$$r_j \vdash setFunctions_{r_j}, cst_{r_j}$$

To receive a significant role, each role should have at least one function assigned.
In UML meta-model, each actor should be assigned at least to one use case in the use case diagram

$$a_j \vdash setUseCases_{a_j}, setConstraints_{a_j}$$

4. A function is defined by a set of permissions necessary to perform such function in accordance with potential security constraints defined for such functions or security constraints defined on the permissions (i.e. authorizations, obligations, conditions)

$$f_k \vdash setPermissions_{f_k}, cst_{f_k}, setPermissionCst_p$$

where

$$setPermissionCst_{p_l} \vdash A_{p_l}, B_{p_l}, C_{p_l}$$

To receive a significant function, each function should have at least one permission assigned. In UML description, using the interaction diagram (in our case sequence diagram) each use case should be defined by detail description, i.e. represented by a set of methods executed on the objects

$$uc_k \vdash setPrivileges_{uc_k}, setConstrains_{uc_k}$$

5.2 Creation of user profiles

The creation process of user profiles, i.e. production of set of roles, in an information system with the use of UML diagrams contains two stages [12]:

- determination of a *set of privileges* (i.e. permissions) for a *use case* in order to define a function and
- determination of a *set of use cases* (i.e. functions) for an *actor* in order to define a role.

In order to define the security profiles for the system's users or groups of users, the set of roles should be assigned to the subject profiles (i.e. user profiles). This task is realized by the security administrator during the exploitation stage. Administrator has to take into consideration the security constraints defined on the global level and subject attributes defined for subjects (i.e. users or groups of users) that determine the access control rights of particular system's users.

6 Conclusion

The extended RBAC is used for managing secure organizations and their functions in information systems. On the other hand, usage control enables dynamic access control during the user access to the information systems by introducing obligations and conditions in access control model. Both role management and usage control are important aspects of access control for modern information systems governed by a certain organization.

Usage Role-based Access Control approach presented above allows to define the access control policy based on access request, as traditional access control models, and the access decision can be evaluated during the access to information to which we want to control the usage. The model takes into consideration the provisional aspects in access security. The components of URBAC can create the framework based on three criteria: functional decision predicates (i.e. authorizations, obligations and conditions), continuity feature (control decision can be taken before or during the access) and mutability feature (updates of subject or object attributes can be done at different time).

The aspects of presented approach are implemented on the software platform that provides to manage the logical security of company information system from the point of view of application developer and from the point of view of security administrator. The platform allows the realization of role engineering process based on URBAC and to allow the management of information system security on access control level.

References

1. R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, Role-Based Access Control Models, IEEE Computer, Volume 29, No 2, s. 38-47, 1996

2. D. Ferraiolo, R. S. Sandhu, S. Gavrila, D. R. Kuhn and R. Chandramouli, Proposed NIST Role-Based Access control, ACM TISSEC, 2001
3. G. Booch, J. Rumbaugh and I. Jacobson, The Unified Modeling Language User Guide, Addison Wesley, 2004
4. OMG Unified Modeling Language (OMG UML): Superstructure. available at: http://www.omg.org/technology/documents/formal/uml.htm, February 2009. Version 2.2, The Object Management Group
5. G.-J. Ahn and R. S. Sandhu, Role-based Authorization Constraints Specification, ACM Transactions on Information and Systems Security, 2000
6. J. Park, X. Zhang and R. Sandhu, Attribute Mutability in Usage Control, 18th IFIP WG 11.3 Working Conference on Data and Applications Security, 2004
7. A. Lazouski, F. Martinelli and P. Mori, Usage control in computer security: A survey, Computer Science Review Vol. 4, Issue 2, May 2010, pp 81-99
8. A. Pretschner , M. Hilty and D. Basin, Distributed usage control, Communications of the ACM, Vol. 49, No 9, September 2006
9. X. Zhang, F. Parisi-Presicce, R. Sandhu and J. Park, Formal Model and Policy Specification of Usage Control, ACM TISSEC, 8(4): 351-387, 2005
10. A. Poniszewska-Maranda, G. Goncalves and F. Hemery, Representation of extended RBAC model using UML language, SOFSEM 2005, LNCS 3381, 2005
11. A. Poniszewska-Maranda, Access Control Coherence of Information Systems Based on Security Constraints, LNCS 4166, pages 412-425, Publisher: Springer-Verlag Heidelberg (2006)
12. G. Goncalves and A. Poniszewska-Maranda, Role engineering: from design to evaluation of security schemas, Journal of Systems and Software, Elsevier, No 8, Vol 81, pp 1306-1326, 2008
13. A. Poniszewska-Maranda, Conception Approach of Access Control in Heterogeneous Information Systems using UML, Journal of Telecommunication Systems, Springer-Verlag Heidelberg, Vol. 45, No 2-3, pages 177-190, October 2010
14. M. Strembeck and G. Neumann, An Integrated Approach to Engineer and Enforce Context Constraints in RBAC Environments, ACM Trans. Information and System Security, vol. 7, no. 3, 2004, pp. 392427
15. S. Castaro, M. Fugini, G. Martella and P. Samarati, Database Security, Addison-Wesley, 1994
16. E. Bertino, E. Ferrari and V. Atluri, The Specification and Enforcement of Authorization Constraints in Workflow Management Systems. ACM Transactions on Information and System Security (TISSEC), 2(1), February 1999
17. D. Dows, J. Rub, K. Kung and C. Jordan, Issues in discretionary access control, Proc. of IEEE Symposium on Research in Security and Privacy, pp 208-218, 1985
18. E. Bertino, C. Bettini and P. Samarati, Temporal Access Control Mechanism for Database Systems, IEEE Trans. on Knowledge and Data Engineering, No 8, 1996
19. E. Bertino, P. Bonatti and E. Ferrari, A Temporal Role-based Access Control Model, ACM Trans. on Information and System Security, No 4(3), pp 191-233, 2001
20. A. Gal and V. Atluri, An Authorization Model for Temporal Data, ACM Transaction on Information and System Security, No 5(1), 2002
21. B. James, E. Joshi, U. Bertino, A. Latif and A. Ghafoo, A Generalized Temporal Role-Based Access Control Model, IEEE Transitions on Knowledge and Data Engineerin, No 17(1), pp 4-23, 2005
22. A. Poniszewska-Maranda, Implementation of Access Control Model for Distributed Information Systems using Usage Control, P. Bouvry et al. (Eds.): SIIS 2011, LNCS 7053, pages 54-67, Publisher: Springer, Heidelberg (2011)